Homework 9 Introduction to Big Data Systems

Christoffer Brevik

November 20, 2024

1 Code overview

To solve this weeks assignment I made 2 files with a C++ program utilizing GridGraph. My code is stored in the files `kcores.cpp` and `pagerank_delta.cpp`.

1.0.1 Running the Script

Running my Partitioning program can be done by going into the `./2024403421/hw9_src` folder in the computational node, or from the root folder of my delivery file. From here you will navigate to the Gridgraph folder, compile the code, preprocess the dataset and run my code by using the following commands:

1.0.2 Navigate to hw9_GridGraph

All my code is located in this folder (more specifically the /mycode sub-folder). You access it trough running the command:

```
cd hw9_GridGraph
```

1.0.3 Makefile

This code will compile all the code, both mine and the example code. I've already changed the Makefile, and pre-cleaned it for you by running `make clean`. So you just have to run:

```
make
```

This may display a lot of warnings in the console, but these were already here from when I opened the assignment. And as it was outside the scope of the assingment, I did not try to solve them

1.0.4 Preprocess LiveJournal

This code will compile all the code in the delievery. In the terminal you run:

```
bin/preprocess -i /data/hw9_data/livejournal -o processed/LiveJournal_Grid -v 4847571 -p 4 -t 0
```

1.0.5 Running the algoirthms

Now that the LiveJournal graph is processed, we can run my programs. Both the codes and the following commands are also explained in the next chapter. To run either of my alogirthms you run:

for K-Cores:

```
bin/kcores processed/LiveJournal_Grid 20 8
```

For PageRank-Delta:

```
bin/pagerank_delta processed/LiveJournal_Grid 20 8
```

2 My code

To solve the assignment I made two C++ files residing in the `mycode` folder. As each file is 80-100 lines I will not post them here, but I will give a brief explanation of each algorithm.

2.1 K-Cores

The K-Cores algorithm identifies the maximal subgraph in which every vertex has at least `k` neighbors. This is achieved by iteratively pruning vertices with a degree less than `k`.

In my implementation, the algorithm starts by calculating the degree of each vertex in the graph and marking vertices as "active" if their degree is greater than or equal to `k`. It then iterates through the graph edges, decrementing the degree of neighbors for any vertex that becomes inactive during a pruning step. The process repeats until no active vertices are left to prune. The remaining vertices form the K-core subgraph.

As stated in Chapter 1, you can run this algorithm by using the command:

```
bin/kcores [path][k] [memory budget]
```

Where:

- [**path**] is the location of our preprocessed graph
- [**k**] is the number of nodes we are filtering by
- [**memory budget**] is the GB that this program can allocate

A recommended setup for this code, with 20 cores is:

```
bin/kcores processed/LiveJournal_Grid 20 8
```

2.2 PageRank Delta

The PageRank Delta algorithm computes the importance of each vertex in a graph using an iterative, delta-based optimization approach. The delta approach incrementally propagates changes in PageRank values, significantly improving computational efficiency compared to standard PageRank implementations.

Initially, each vertex's PageRank is set to a default value (random teleportation factor), and its "delta" is distributed across its outgoing edges. In each iteration, the algorithm propagates the delta values to the neighbors and accumulates these changes in the PageRank values. Once all changes are accounted for, new delta values are calculated for the next iteration. This continues for a fixed number of iterations or until convergence.

The implementation takes advantage of GridGraph's edge streaming and locking mechanisms to handle large-scale data efficiently while ensuring correctness. The damping factor, teleportation factor, and proper handling of zero-degree nodes are incorporated to adhere to the PageRank model.

As stated in Chapter 1, you can run this algorithm by using the command:

```
bin/pagerank_delta [path][iterations] [memory budget]
```

Where:

- [**path**] is the location of our preprocessed graph
- [**iterations**] is the number of iterations that the program will run on
- [**memory budget**] is the GB that this program can allocate

A recommended setup for this code, with 20 iterations is:

```
bin/pagerank_delta processed/LiveJournal_Grid 20 8
```

3 Results

3.1 K-Cores Results

The K-Cores algorithm outputs the number of vertices that belong to the largest `k`-core subgraph. This result is displayed in the terminal . Running it on the dataset you should see:

```
...
* Several Debug messages *
...
Iteration 1: 932824 active vertices
Iteration 2: 11599985 active vertices
K-core (20-core) decomposition complete: 932824 vertices remain
```

3.2 PageRank Delta Results

The PageRank Delta algorithm outputs the computed PageRank values for all vertices in the graph. These values are saved with the files `degree, pagerank, delta and new_delta pagerank`, located in the `processed/` folder (with the preprocessed graph).

While running the algorithm, the terminal displays the progress of the computation, including the time taken for each iteration and the total execution time. Running it on the LiveJournal dataset you should get:

```
...
* Several Debug messages *
...
20 iterations of pagerank delta took 10.30 seconds
```